Fig. 1

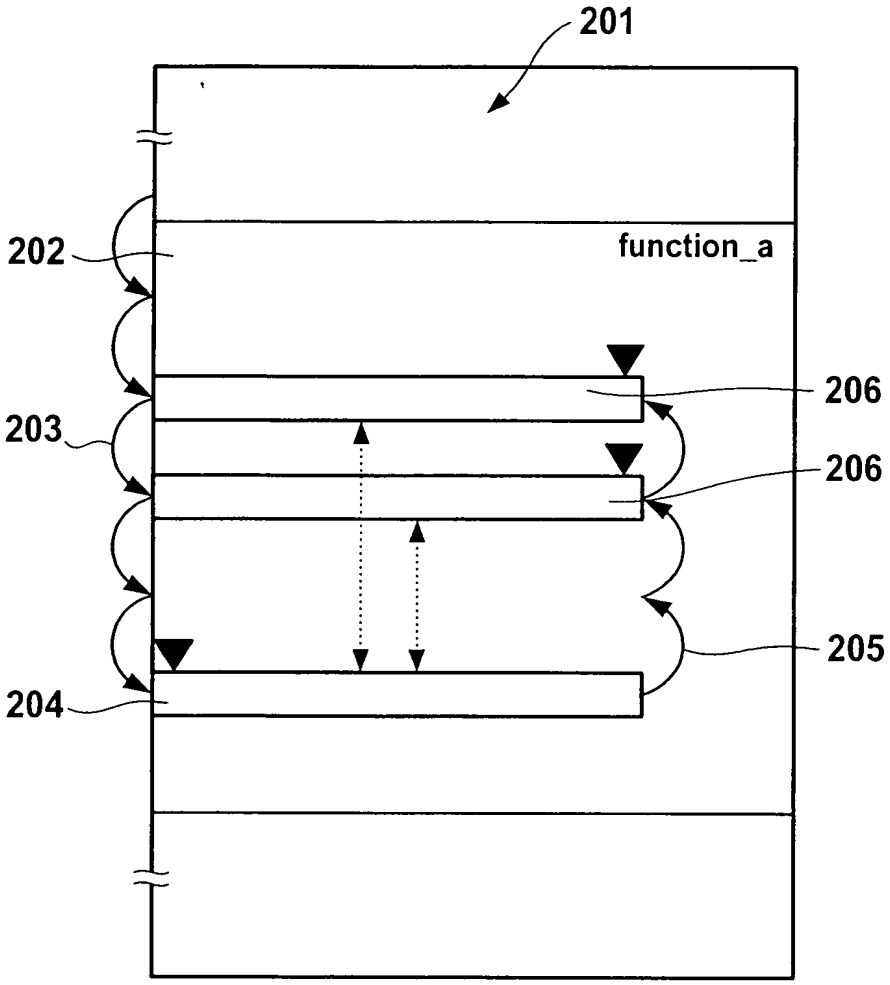


Fig. 2

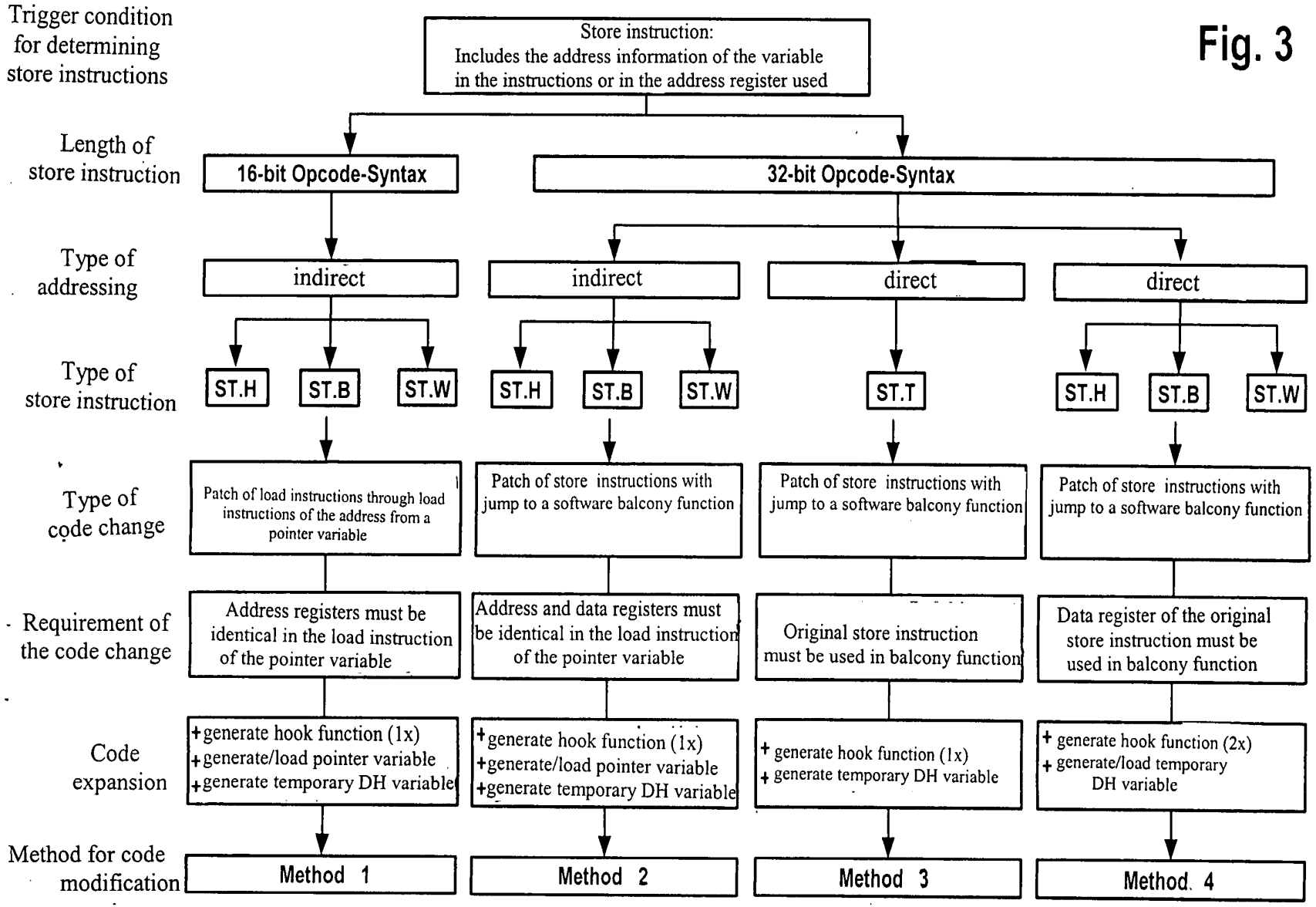

Trigger condition for determining store instructions
Store instruction:
Includes the address information of the variable in the instructions or in the address register used
Length of store instruction
16-bit Opcode-Syntax
32-bit Opcode-Syntax
Type of addressing
indirect
indirect
direct
direct
Type of store instruction
ST.H
ST.B
ST.W
ST.H
ST.B
ST.W
ST.T
ST.H
ST.B
ST.W
Type of code change
Patch of load instructions through load instructions of the address from a pointer variable
Patch of store instructions with jump to a software balcony function
Patch of store instructions with jump to a software balcony function
Patch of store instructions with jump to a software balcony function
Requirement of the code change
Address registers must be identical in the load instruction of the pointer variable
Address and data registers must be identical in the load instruction of the pointer variable
Original store instruction must be used in balcony function
Data register of the original store instruction must be used in balcony function
Code expansion
+generate hook function (1x)
+generate/load pointer variable
+generate temporary DH variable
+generate hook function (1x)
+generate/load pointer variable
+generate temporary DH variable
+ generate hook function (1x)
+ generate temporary DH variable
+ generate hook function (2x)
+ generate/load temporary DH variable
Method for code modification
Method 1
Method 2
Method 3
Method 4

Fig. 3

Fig. 4

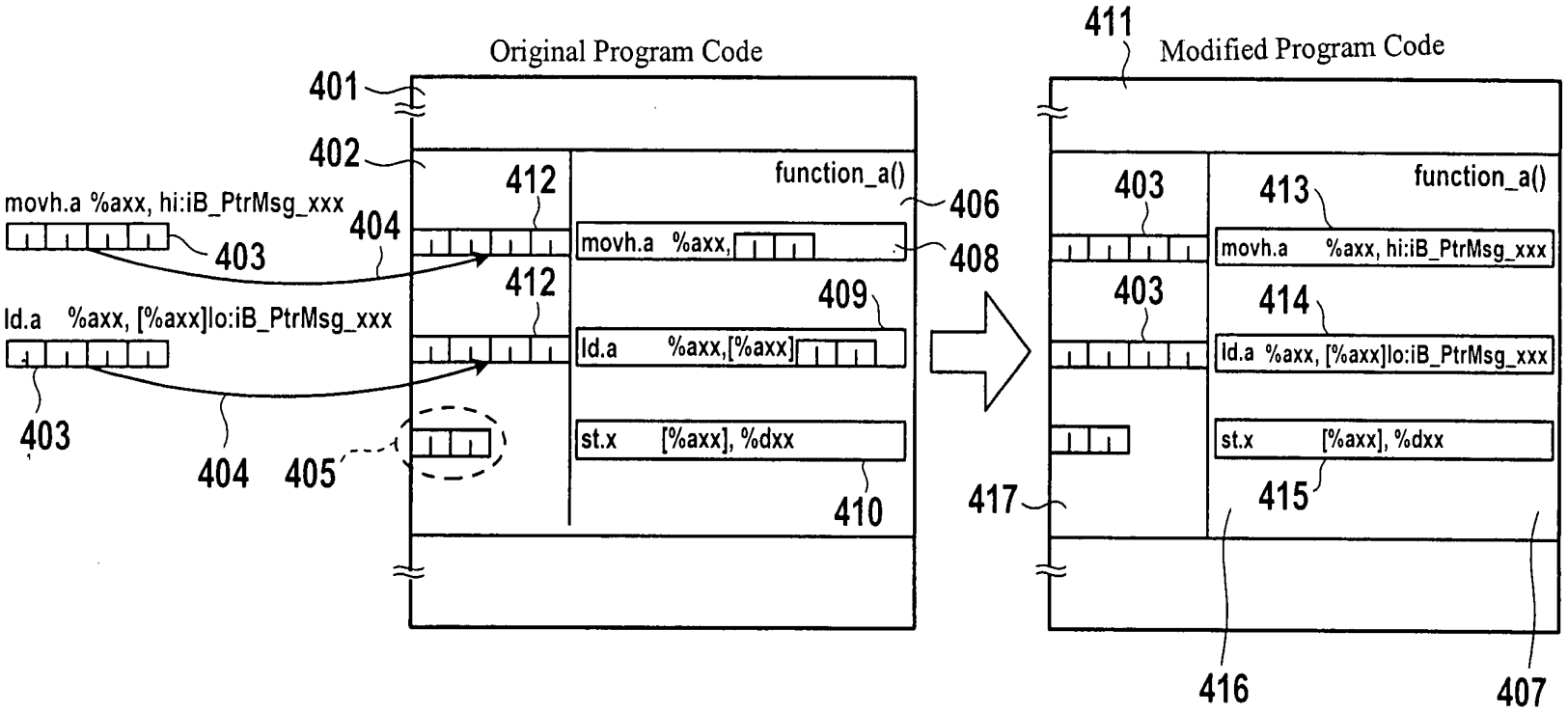
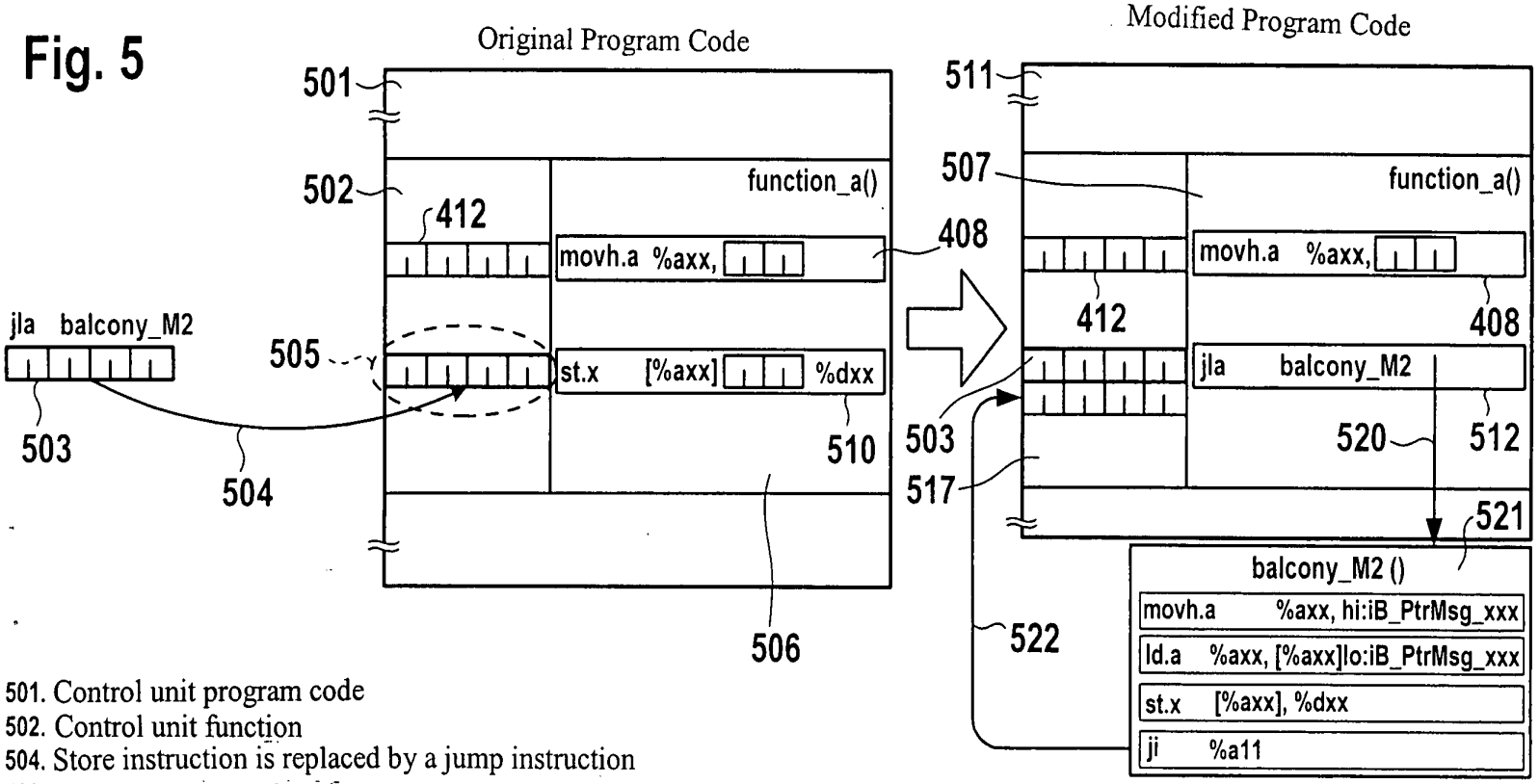

Original Program Code
Modified Program Code
401
402
411
412
function_a()
406
movh.a %axx, hi:iB_PtrMsg_xxx
403
404
movh.a %axx,
408
412
409
ld.a %axx, [%axx]lo:iB_PtrMsg_xxx
ld.a %axx,[%axx]
403
404
405
st.x [%axx], %dxx
410
403
413
function_a()
movh.a %axx, hi:iB_PtrMsg_xxx
403
414
ld.a %axx, [%axx]lo:iB_PtrMsg_xxx
st.x [%axx], %dxx
417
415
416
407

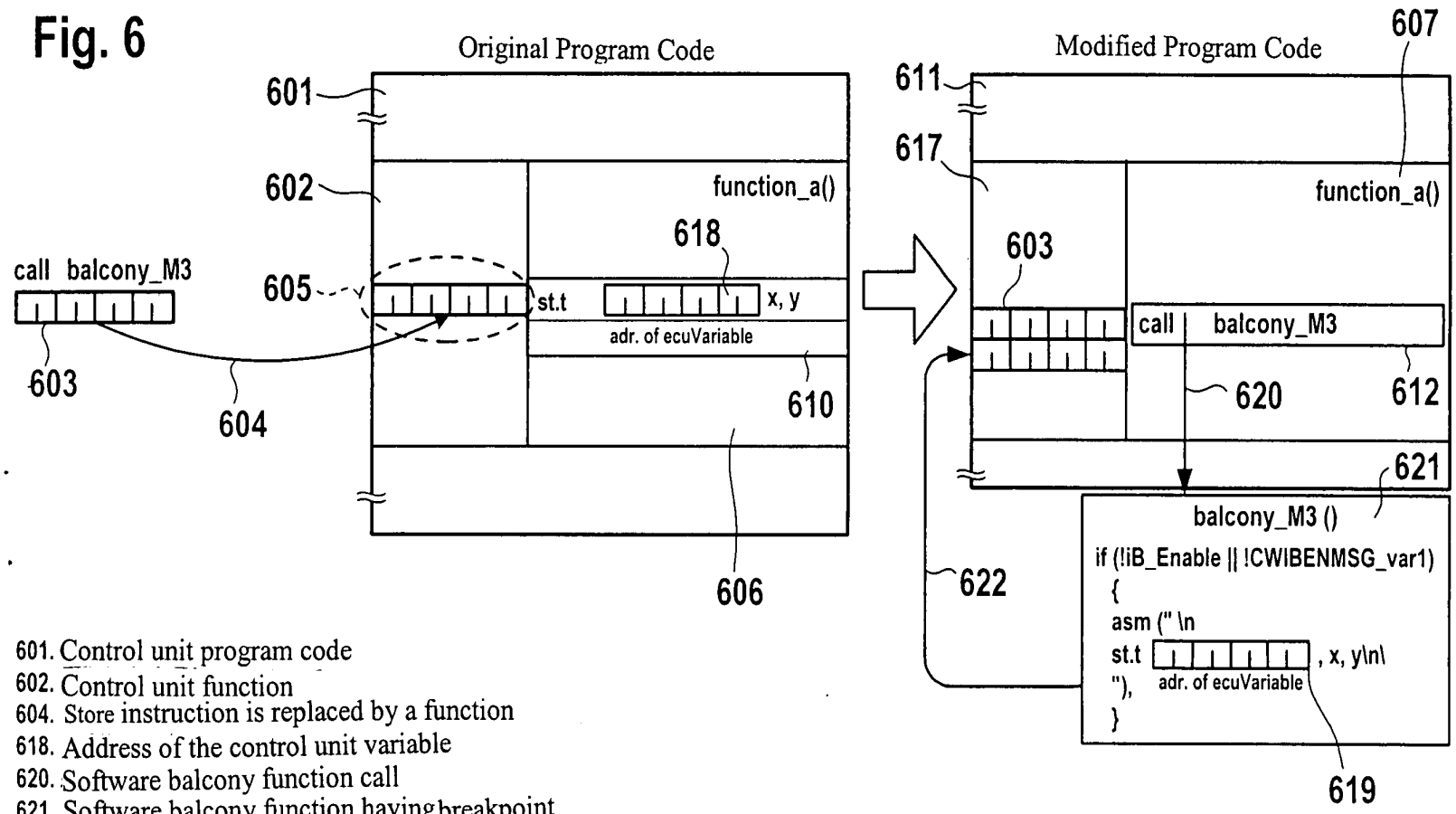
Fig. 5



501. Control unit program code
502. Control unit function
504. Store instruction is replaced by a jump instruction
520. Software balcony function call
521. Software balcony function (diverting the store instruction to pointer variable)
522. Jump back to control unit function

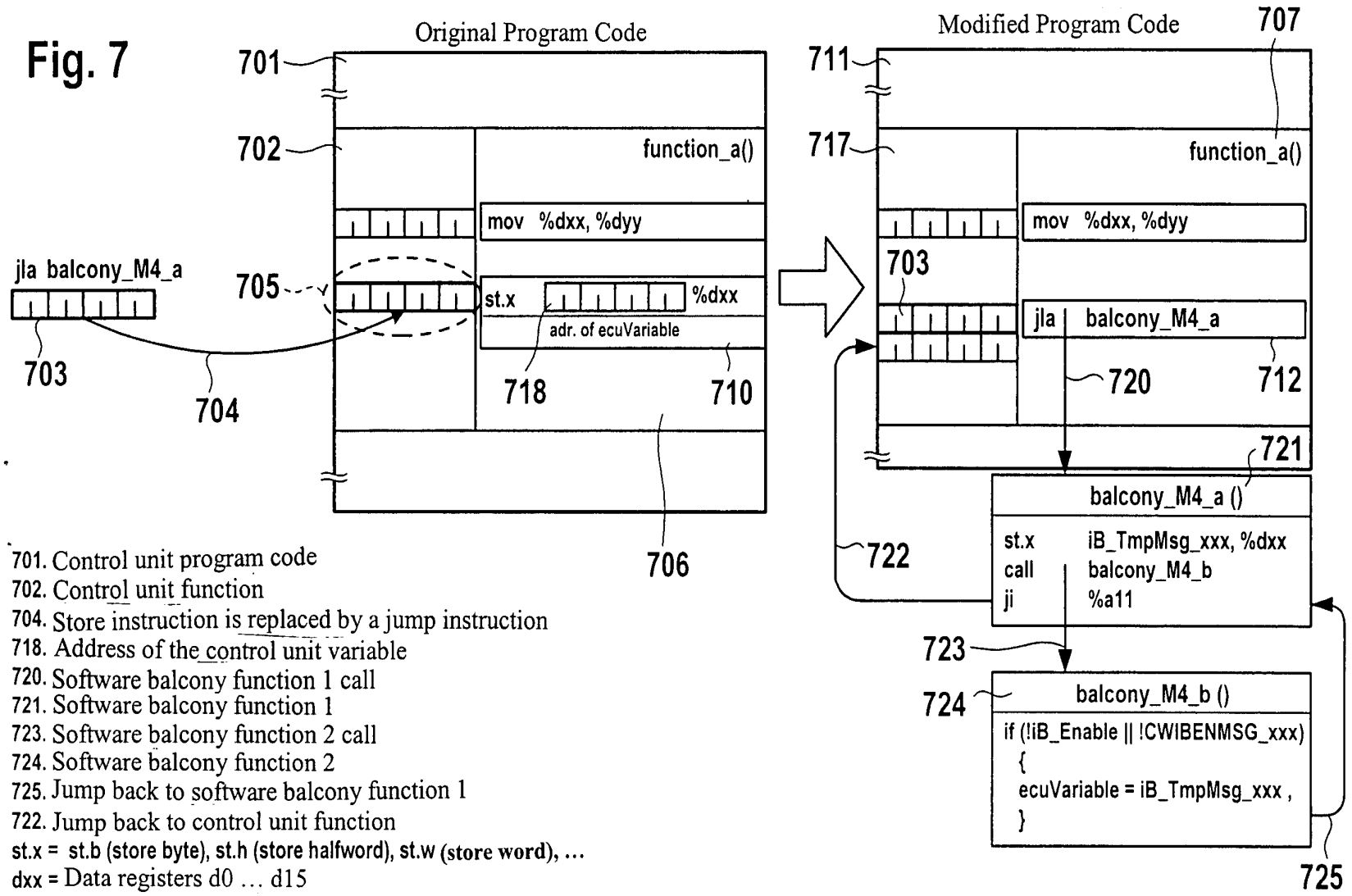
st.x = st.b (store byte), st.h (store halfword), st.w (store word), ...
axx = Address register a0 ... a15
dxx = Data registers d0 ... d15
503. ▭ New program code

Fig. 6



601. Control unit program code
602. Control unit function
604. Store instruction is replaced by a function
618. Address of the control unit variable
620. Software balcony function call
621. Software balcony function having breakpoint
619. Address of the control unit variable
622. Jump back to control unit function
603. ▭ New program code

Fig. 7



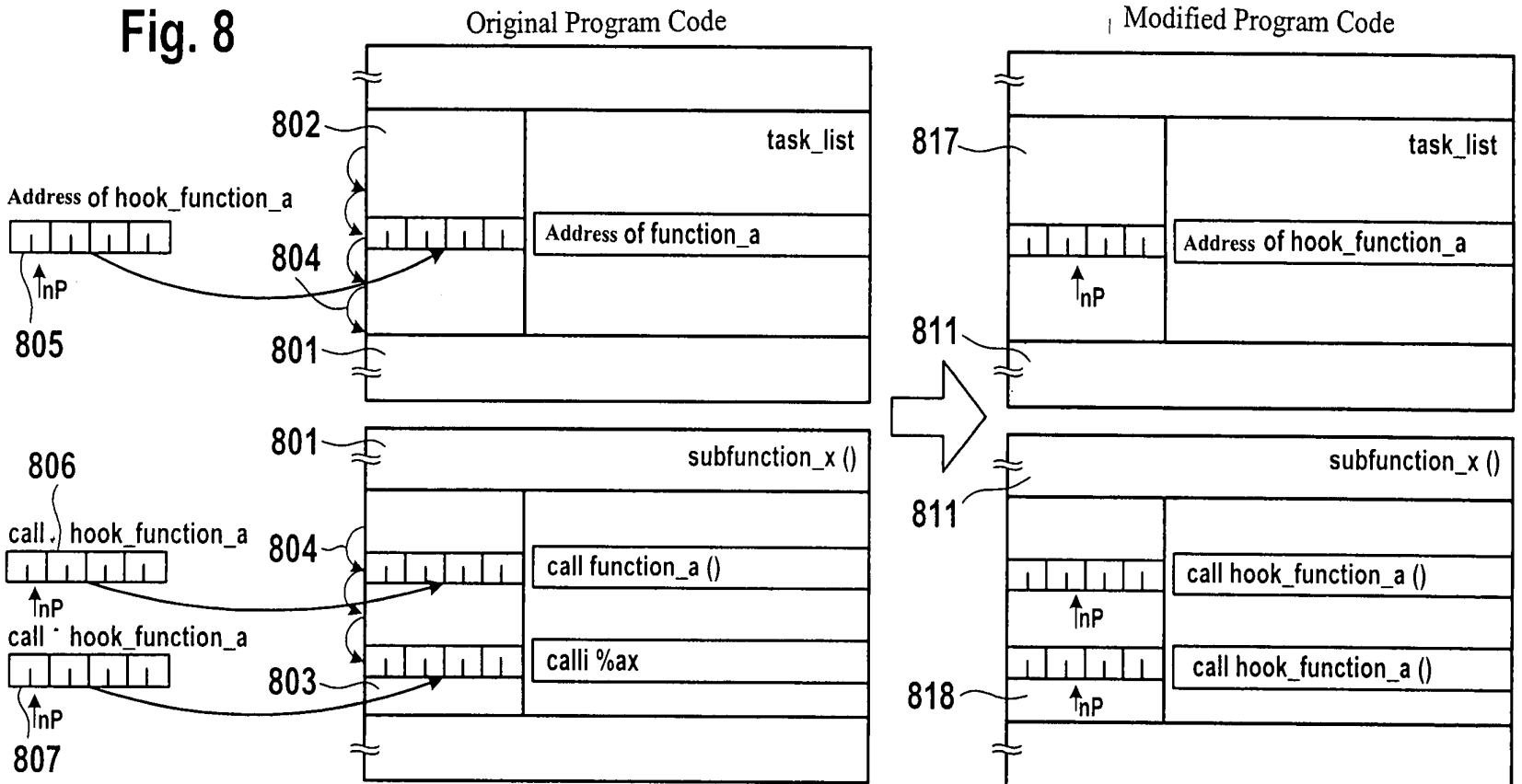
701. Control unit program code
702. Control unit function
704. Store instruction is replaced by a jump instruction
718. Address of the control unit variable
720. Software balcony function 1 call
721. Software balcony function 1
723. Software balcony function 2 call
724. Software balcony function 2
725. Jump back to software balcony function 1
722. Jump back to control unit function

st.x = st.b (store byte), st.h (store halfword), st.w (store word), ...

dxx = Data registers d0 ... d15

703. ▭ New program code

Fig. 8



801. Control unit program code
802. Task list
803. Subfunction
804. Procedure for determination of function addresses and function calls
805. Address of function_a is replaced by address of hook_function_a
806. Function call of function_a is replaced by call of hook_function_a
807. Indirect function call of function_a is replaced by call of hook_function_a (32-bit instruction)

nP ▭ New program code

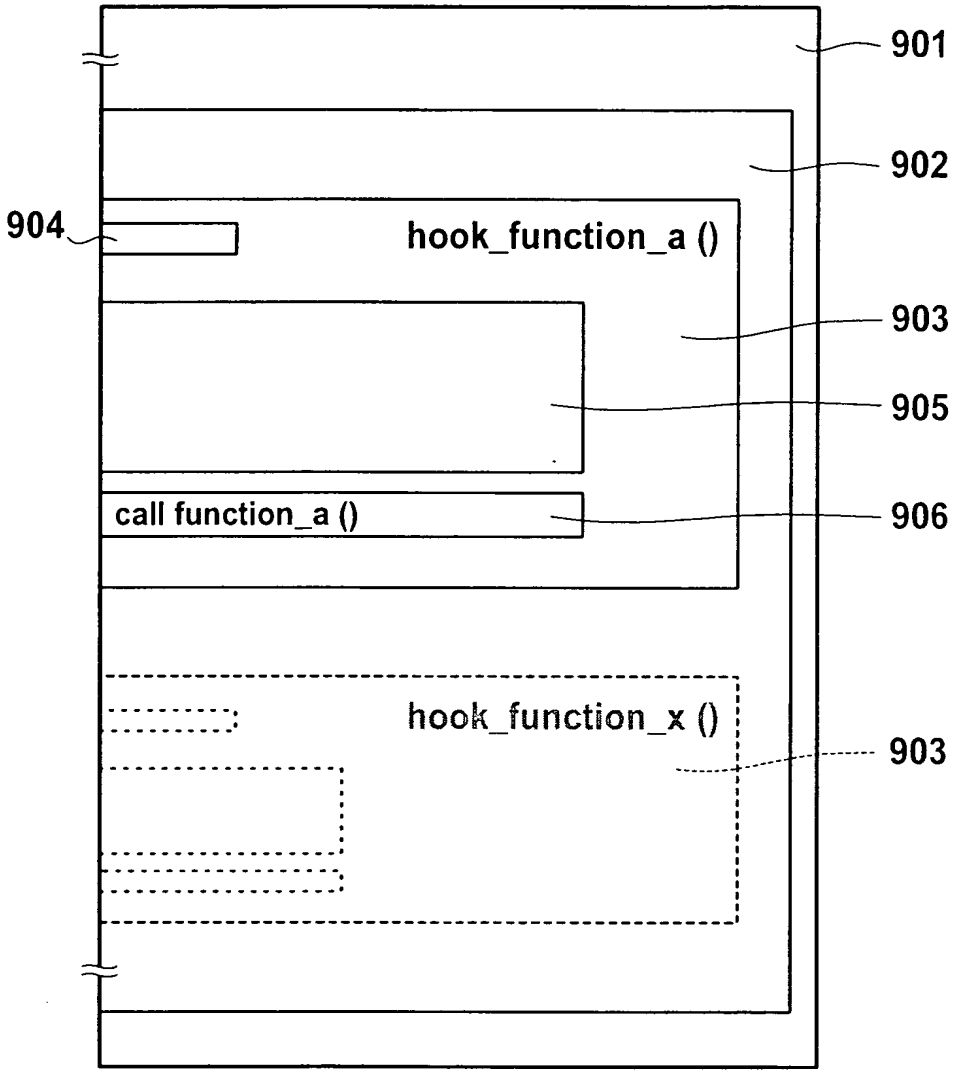

901
902
904
hook_function_a ()
903
905
call function_a ()
906
hook_function_x ()
903

Fig. 9

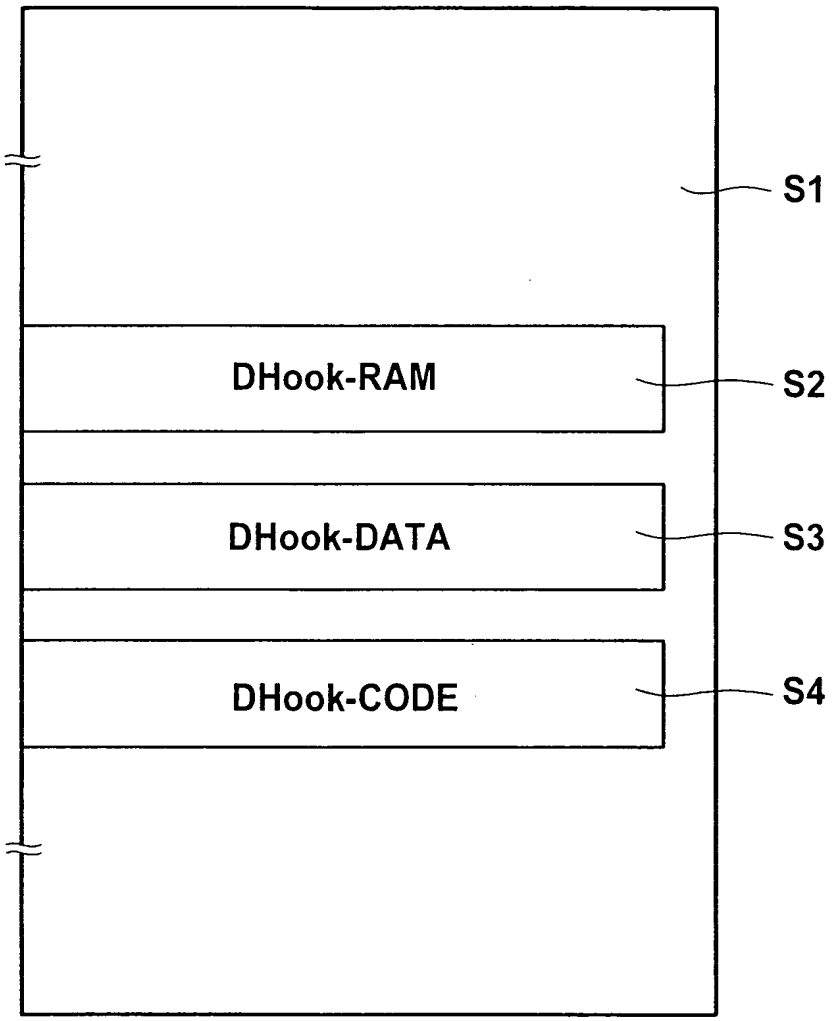


Fig. 10

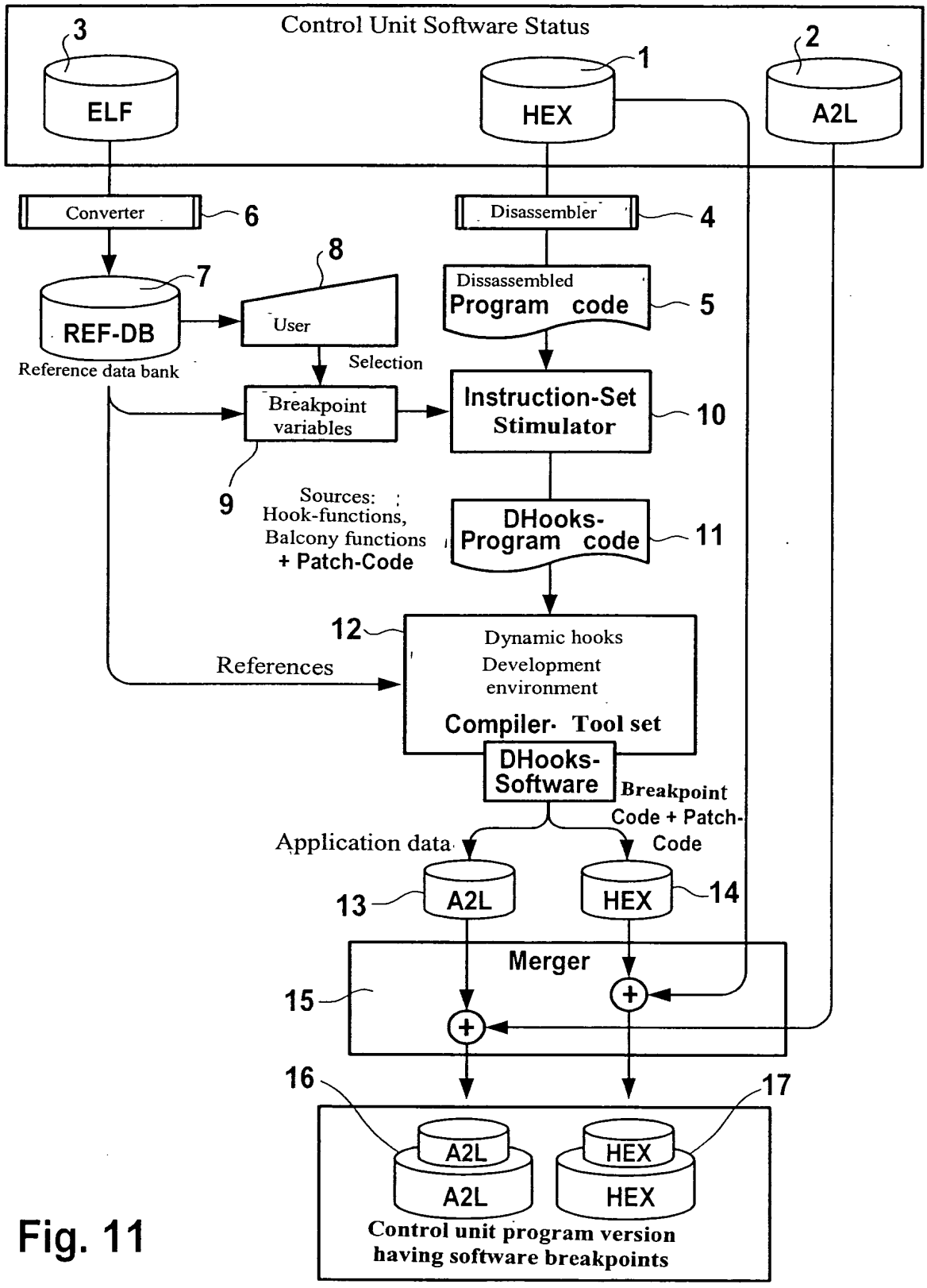

Control Unit Software Status
3
ELF
1
HEX
2
A2L
Converter
6
Disassembler
4
8
7
REF-DB
Reference data bank
User
Selection
Dissassembled
Program code
5
Breakpoint
variables
Instruction-Set
Stimulator
10
9
Sources:
Hook-functions,
Balcony functions
+ Patch-Code
DHooks-
Program code
11
12
Dynamic hooks
Development
environment
References
Compiler- Tool set
DHooks-
Software
Breakpoint
Code + Patch-
Code
Application data
13
A2L
HEX
14
Merger
15
16
17
A2L
A2L
HEX
HEX
Control unit program version
having software breakpoints

Fig. 11

Fig. 12

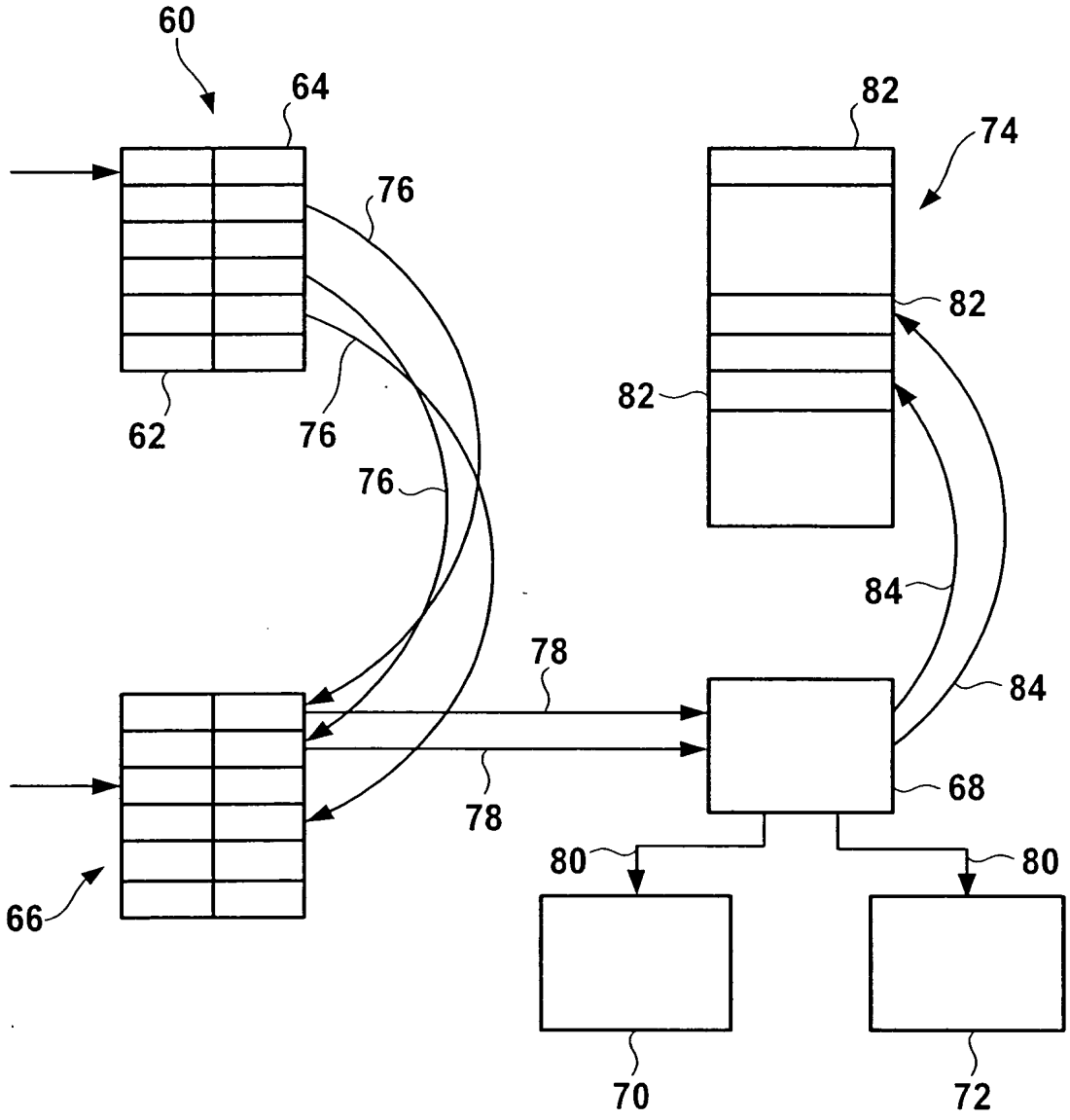
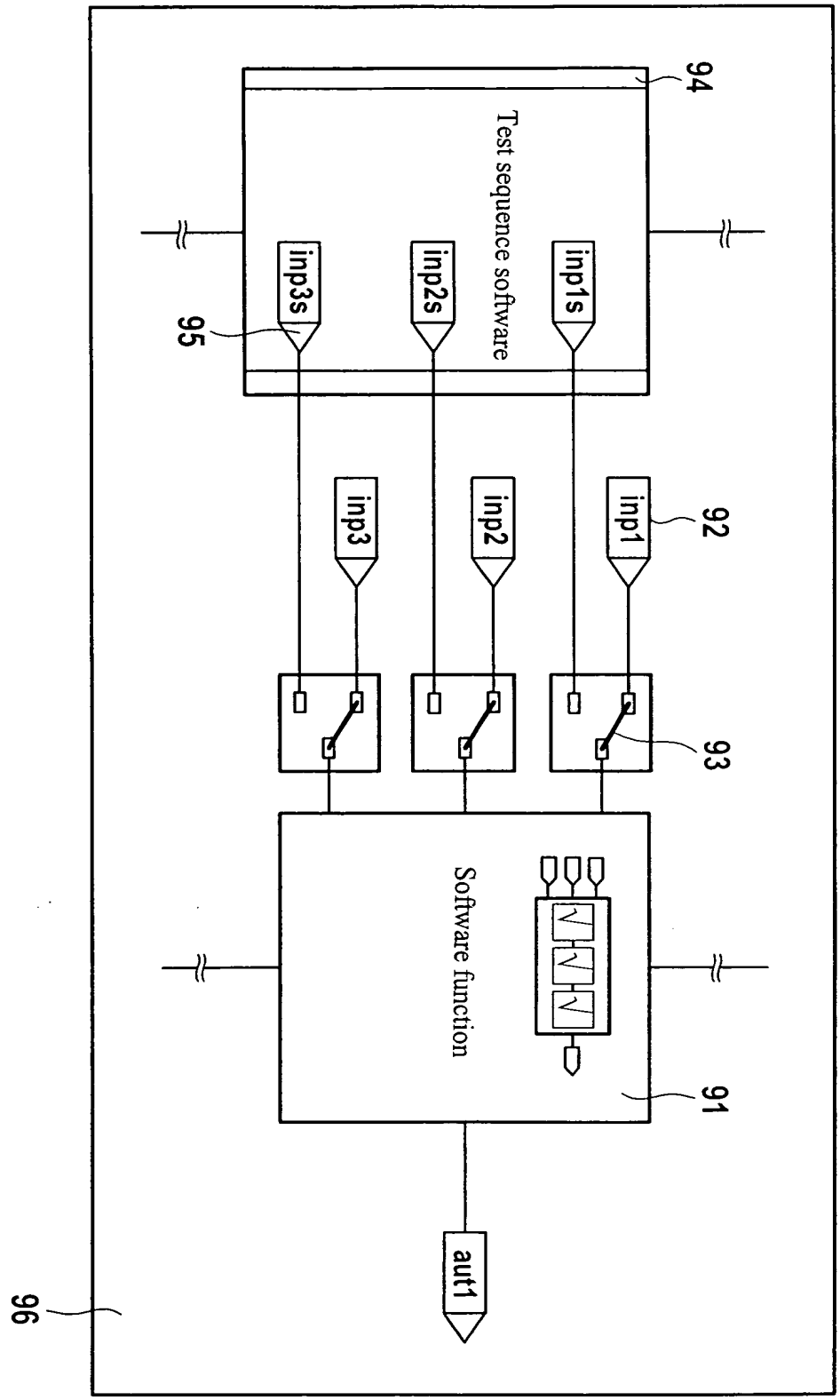


Fig. 13


94
Test sequence software
inp3s
inp2s
inp1s
95
inp3
inp2
inp1
92
93
Software function
91
aut1
96

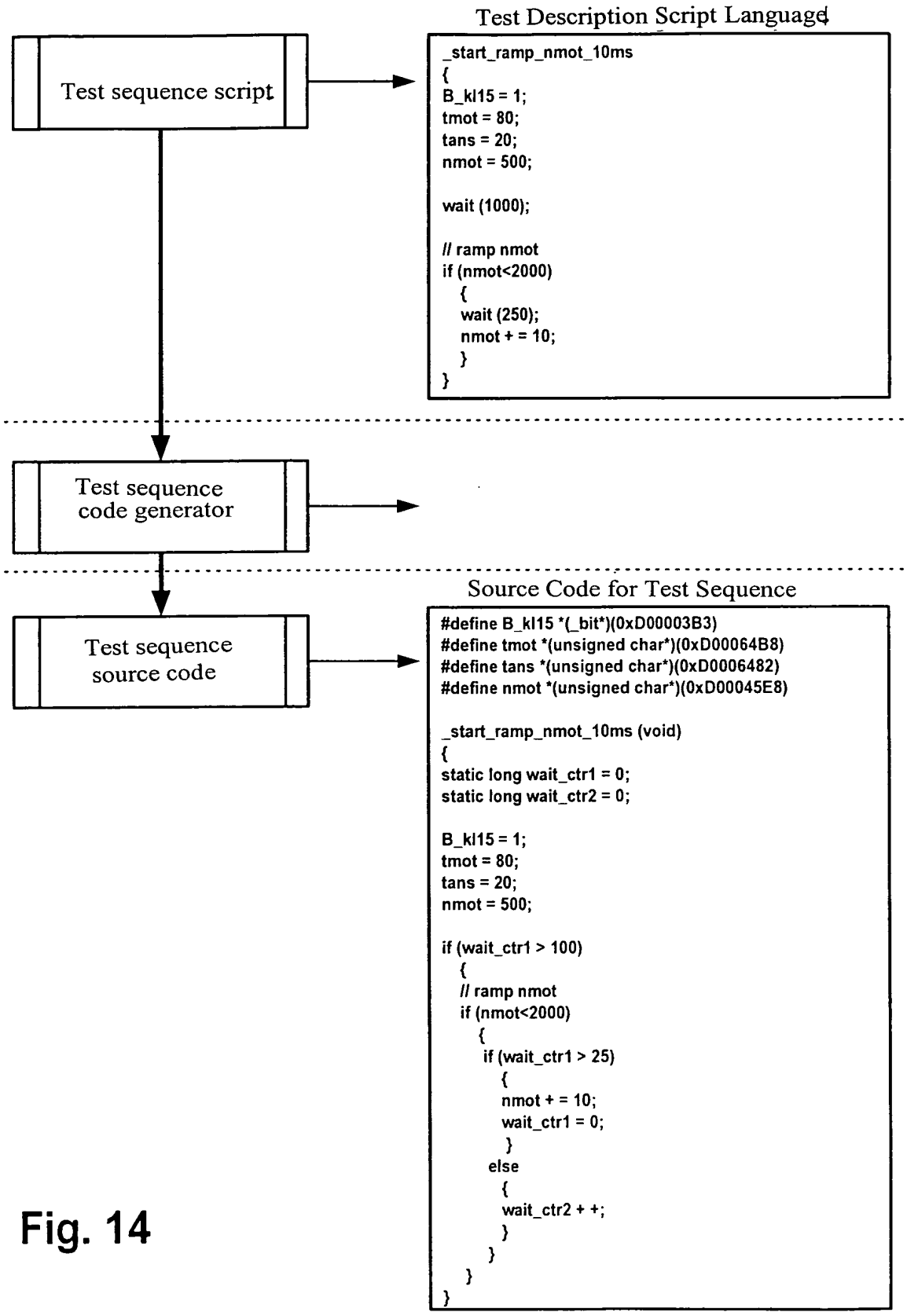
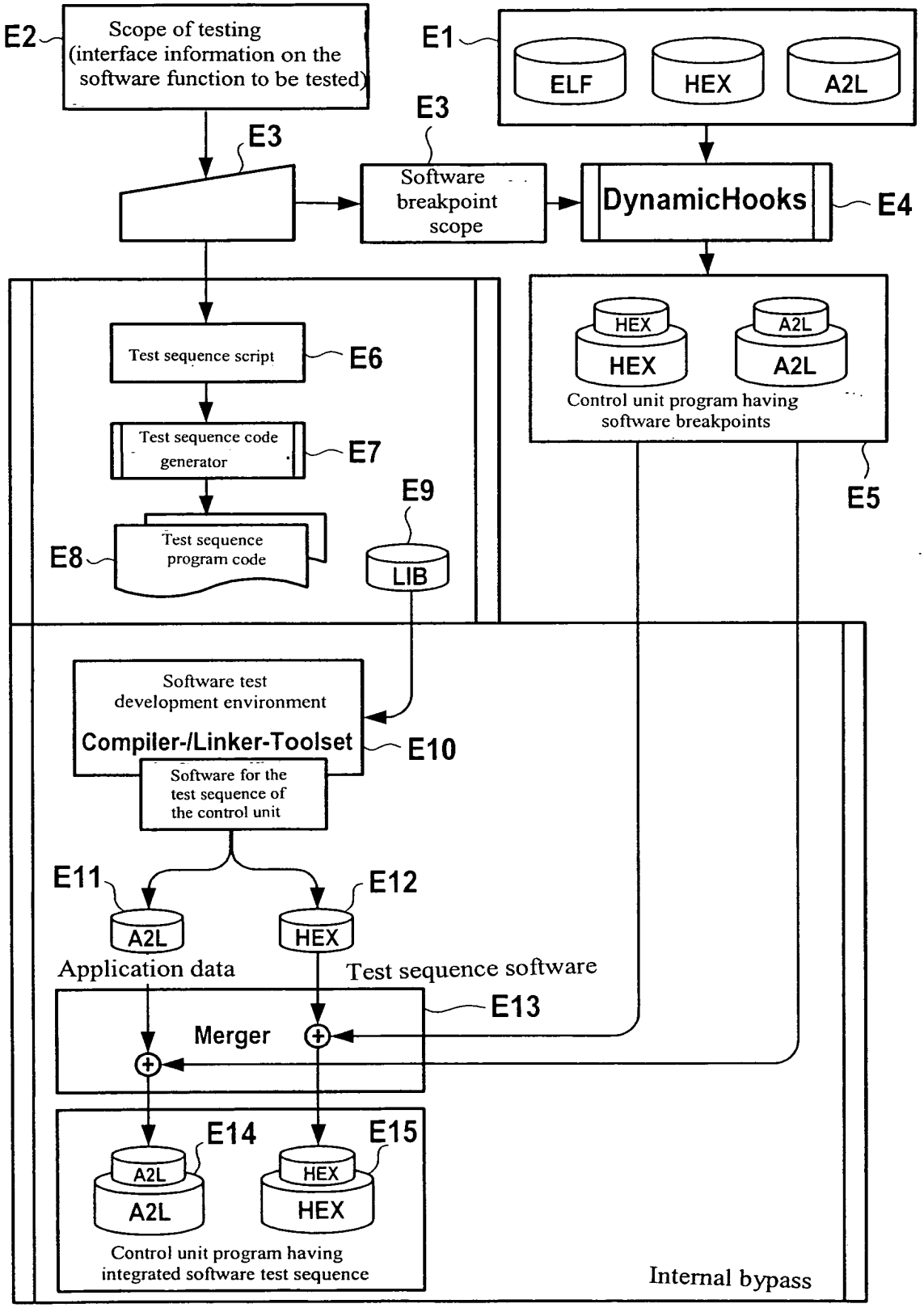

Test Description Script Language
Test sequence script
_start_ramp_nmot_10ms
{
B_kl15 = 1;
tmot = 80;
tans = 20;
nmot = 500;
wait (1000);
// ramp nmot
if (nmot<2000)
{
wait (250);
nmot + = 10;
}
}
Test sequence code generator
Source Code for Test Sequence
Test sequence source code
#define B_kl15 *(_bit*)(0xD00003B3)
#define tmot *(unsigned char*)(0xD00064B8)
#define tans *(unsigned char*)(0xD0006482)
#define nmot *(unsigned char*)(0xD00045E8)
_start_ramp_nmot_10ms (void)
{
static long wait_ctr1 = 0;
static long wait_ctr2 = 0;
B_kl15 = 1;
tmot = 80;
tans = 20;
nmot = 500;
if (wait_ctr1 > 100)
{
// ramp nmot
if (nmot<2000)
{
if (wait_ctr1 > 25)
{
nmot + = 10;
wait_ctr1 = 0;
}
else
{
wait_ctr2 + +;
}
}
}
}

Fig. 14

Fig. 15


E2
Scope of testing (interface information on the software function to be tested)
E1
ELF
HEX
A2L
E3
E3
Software breakpoint scope
DynamicHooks
E4
Test sequence script
E6
Test sequence code generator
E7
E9
E8
Test sequence program code
LIB
HEX
HEX
A2L
A2L
Control unit program having software breakpoints
E5
Software test development environment
Compiler-/Linker-Toolset
E10
Software for the test sequence of the control unit
E11
E12
A2L
HEX
Application data
Test sequence software
E13
Merger
E14
E15
A2L
A2L
HEX
HEX
Control unit program having integrated software test sequence
Internal bypass